

Matrix-Accelerated Key Aggregation for Reducing ECDSA Cracking Complexity: A Theoretical Framework

Harsh Patel

Keywords: ECDSA · Elliptic Curve Cryptography · Matrix Multiplication · Quantum Cryptanalysis · Harvest Now Decrypt Later · Key Aggregation

Abstract

The Elliptic Curve Digital Signature Algorithm (ECDSA) underpins the security of modern digital infrastructure—from TLS certificates to blockchain transactions. While classically intractable, ECDSA is known to be vulnerable to quantum adversaries via Shor’s algorithm, which can solve the Elliptic Curve Discrete Logarithm Problem (ECDLP) in polynomial time. However, a critical and under-examined constraint in real-world quantum attack scenarios is the *batch problem*: an adversary engaged in a *Harvest Now, Decrypt Later* (HNDL) campaign must crack not one, but potentially millions of independent public keys.

This paper proposes a novel theoretical framework—**Matrix-Accelerated Key Aggregation (MAKA)**—that leverages structured matrix multiplication over elliptic curve point arithmetic to algebraically combine multiple unknown public keys with known curve points into a single *composite key*. By inverting the aggregation after cracking only the composite key, the adversary recovers all constituent private keys simultaneously, thereby reducing the effective quantum computational complexity from $\mathcal{O}(n \cdot \text{poly}(\log q))$ to $\mathcal{O}(\text{poly}(\log q) + n \cdot M(n))$, where n is the number of target keys, q is the curve order, and $M(n)$ is the cost of matrix inversion.

We formalise the construction, analyse its algebraic validity over elliptic curve groups, discuss the conditions under which the aggregation is invertible, and examine the cryptographic and defensive implications of this approach.

1 Introduction

Elliptic Curve Cryptography (ECC) has been the cornerstone of public-key cryptography for over two decades. Its appeal lies in offering equivalent security to RSA at significantly smaller key sizes—a 256-bit elliptic curve key provides roughly the same security as a 3072-bit RSA key. The Elliptic Curve Digital Signature Algorithm (ECDSA) is the most widely deployed ECC-based signature scheme, securing everything from HTTPS connections to Bitcoin transactions.

The advent of quantum computing poses an existential threat to ECDSA. Shor’s algorithm, when executed on a sufficiently powerful quantum computer, can solve the Elliptic Curve Discrete Logarithm Problem (ECDLP) in polynomial time, effectively breaking ECDSA entirely [1]. While large-scale fault-tolerant quantum computers do not yet exist, the **Harvest Now, Decrypt Later (HNDL)** attack model presents an immediate and practical concern: adversaries can intercept and store encrypted communications today with the intent of decrypting them once quantum capabilities mature [2].

A critical observation—and the motivating insight for this paper—is that a real-world HNDL adversary does not face a *single* ECDLP instance. Instead, they accumulate a vast corpus of intercepted data protected by *many different* ECDSA public keys. Naïvely, each key requires an independent execution of Shor’s algorithm, making the total quantum cost scale linearly with the number of keys.

This paper asks a fundamental question:

Can we algebraically pre-process a batch of ECDSA public keys—combining them with known elliptic curve points via matrix multiplication—to produce a single composite key whose cracking reveals all constituent private keys?

We answer this question affirmatively under specific algebraic conditions and present the **Matrix-Accelerated Key Aggregation (MAKA)** framework. MAKA operates entirely in the classical domain as a pre-processing step, restructuring the cryptanalytic workload so that the expensive quantum phase is invoked only once, regardless of the number of target keys.

2 Background and Preliminaries

2.1 Elliptic Curve Cryptography and ECDSA

An elliptic curve E over a finite field $\mathbb{F}_p$ (where p is a large prime) is defined by an equation of the form

$$y^2 \equiv x^3 + ax + b \pmod{p}, \quad (1)$$

where $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$ (to ensure non-singularity). The set of points on this curve, together with a special *point at infinity* $\mathcal{O}$, forms an abelian group under a geometrically-motivated addition operation.

In ECDSA a standard base point (generator) G of prime order q is fixed as a public parameter. Key generation proceeds as follows:

- **Private key:** Select a random integer d with $1 \leq d \leq q - 1$.
- **Public key:** Compute $Q = d \cdot G$ (scalar multiplication of the generator).

The security of ECDSA rests on the assumption that recovering d from Q and G —the **Elliptic Curve Discrete Logarithm Problem (ECDLP)**—is computationally infeasible.

2.2 The Elliptic Curve Discrete Logarithm Problem

Definition 2.1 (ECDLP). *Given an elliptic curve E over $\mathbb{F}_p$, a generator point G of order q , and a point $Q \in \langle G \rangle$, find the integer d such that $Q = d \cdot G$.*

The best known classical algorithms for ECDLP are:

- **Baby-step Giant-step:** $\mathcal{O}(\sqrt{q})$ time and space.
- **Pollard’s rho:** $\mathcal{O}(\sqrt{q})$ time, $\mathcal{O}(1)$ space.
- **Pohlig–Hellman:** Efficient when q has small factors (avoided by choosing q prime).

For a 256-bit curve (e.g. `secp256k1`), these algorithms require approximately 2^{128} group operations—considered infeasible with classical hardware.

2.3 Quantum Threats to ECDSA

Shor’s algorithm [1] solves the discrete logarithm problem (including ECDLP) in $\mathcal{O}(\text{poly}(\log q))$ time on a quantum computer, specifically requiring

$$\mathcal{O}((\log q)^2 (\log \log q) (\log \log \log q))$$

quantum gate operations using fast multiplication. For a 256-bit curve this reduces the attack complexity from $\sim 2^{128}$ to roughly 2^{30} —a catastrophic reduction.

The quantum resource requirements, however, remain substantial. Current estimates suggest that breaking a single 256-bit ECDSA key requires approximately **2 330 logical qubits** and 1.26×10^{10} Toffoli gates [3], translating to millions of physical qubits with current error-correction overhead.

2.4 The Harvest Now, Decrypt Later Paradigm

The HNDL threat model operates in two phases:

1. **Harvest Phase (Classical, Present):** The adversary intercepts and stores encrypted network traffic, public keys, digital signatures, and key-exchange parameters. This is a passive, low-cost operation.
2. **Decrypt Phase (Quantum, Future):** Once a sufficiently powerful quantum computer becomes available, the adversary processes the harvested data to extract private keys and decrypt stored communications.

The key challenge in the decrypt phase is **throughput**: the adversary may have harvested millions of distinct public keys, each requiring a separate quantum computation. If each quantum ECDLP solution takes time T_q , processing n keys requires total time $n \cdot T_q$.

This linear scaling is the bottleneck that MAKa aims to address.

3 Problem Statement

Given:

- An elliptic curve $E(\mathbb{F}_p)$ with generator G of prime order q .
- A set of n target ECDSA public keys $\{Q_1, Q_2, \dots, Q_n\}$ where $Q_i = d_i \cdot G$.
- A set of m known points on the curve $\{P_1, P_2, \dots, P_m\}$ where $P_j = k_j \cdot G$ and each k_j is known to the adversary.
- Access to a quantum computer capable of solving a single ECDLP instance.

Goal: Construct a classical pre-processing procedure that combines the known points $\{P_j\}$ and unknown public keys $\{Q_i\}$ via matrix multiplication to produce a **single composite point** C on the curve, such that:

1. Solving the ECDLP for C (finding c where $C = c \cdot G$) is sufficient.
2. The scalar c , combined with the known structure of the matrix construction, allows recovery of **all** private keys $\{d_1, d_2, \dots, d_n\}$.

Complexity target: Reduce total cryptanalytic cost from $\mathcal{O}(n \cdot T_q)$ to $\mathcal{O}(T_q + n \cdot T_{\text{classical}})$ where $T_{\text{classical}} \ll T_q$.

4 Matrix-Accelerated Key Aggregation (MAKA)

4.1 Core Intuition

The fundamental insight of MAKa is that elliptic curve point addition is an abelian group operation and scalar multiplication distributes over it. A *linear combination* of public keys can therefore be expressed as a single scalar multiple of the generator:

If $C = \alpha_1 Q_1 + \alpha_2 Q_2 + \dots + \alpha_n Q_n$ where $Q_i = d_i \cdot G$, then

$$C = (\alpha_1 d_1 + \alpha_2 d_2 + \dots + \alpha_n d_n) \cdot G. \quad (2)$$

The composite scalar is $c = \sum_i \alpha_i d_i \pmod{q}$. Cracking C via quantum computation yields c , but a single linear equation in n unknowns is insufficient for recovery.

The matrix approach extends this by constructing n independent linear combinations using both known scalars $\{k_j\}$ and unknown scalars $\{d_i\}$, structuring them as a matrix equation that is invertible.

4.2 Formal Construction

Step 1: Construct the Mixing Matrix. Define an $n \times n$ invertible matrix $\mathbf{A}$ over $\mathbb{Z}_q$ (integers modulo the curve order):

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad (3)$$

where each $a_{ij} \in \mathbb{Z}_q$ and $\det(\mathbf{A}) \not\equiv 0 \pmod{q}$. The entries of $\mathbf{A}$ are chosen by the adversary—they are known quantities.

Step 2: Define the Private Key Vector. Let the vector of unknown private keys be

$$\mathbf{d} = (d_1, d_2, \dots, d_n)^\top. \quad (4)$$

Step 3: Compute the Composite Scalar Vector. The matrix-vector product yields a vector of composite scalars:

$$\mathbf{c} = \mathbf{A} \mathbf{d} \pmod{q}. \quad (5)$$

Explicitly:

$$c_i = \sum_{j=1}^n a_{ij} d_j \pmod{q}, \quad i = 1, 2, \dots, n. \quad (6)$$

Step 4: Compute Composite Points (Classical). Each composite scalar corresponds to a composite point on the curve:

$$C_i = c_i \cdot G = \sum_{j=1}^n a_{ij} \cdot Q_j. \quad (7)$$

These composite points C_i can be computed classically using only the *public keys* $\{Q_j\}$ and the known matrix entries $\{a_{ij}\}$ —no knowledge of private keys is required.

Remark 4.1. *Computing $C_i = \sum_j a_{ij} \cdot Q_j$ requires only elliptic curve point addition and scalar-point multiplication, both of which are efficient classical operations.*

4.3 Aggregation into a Singular Composite Key

The construction in Section 4.2 produces n composite points, each of which would still require a separate quantum ECDLP solution. To achieve the goal of a **single** quantum computation, we apply a *hierarchical aggregation*.

4.3.1 Recursive Aggregation

1. **Level 0:** Start with n composite points $\{C_1^{(0)}, \dots, C_n^{(0)}\} = \{C_1, \dots, C_n\}$.
2. **Level 1:** Choose known random scalars $\beta_1, \dots, \beta_n \in \mathbb{Z}_q$ and compute:

$$C^{(1)} = \sum_{i=1}^n \beta_i C_i^{(0)}. \quad (8)$$

3. **Result:** A single point $C^{(1)} = c^* \cdot G$ where

$$c^* = \beta^\top \mathbf{A} \mathbf{d} \pmod{q}. \quad (9)$$

This produces one composite point but yields only one equation. The critical challenge is recovering n unknowns from fewer equations. Section 4.4 addresses this through a structured approach.

4.3.2 The Structured MAKa Protocol

Rather than collapsing all information into a single scalar, MAKa uses a carefully designed sequence of n aggregations, each structured so that a single quantum ECDLP solution provides maximal information.

Phase 1—Structured Injection of Known Points. Construct n composite points where each incorporates a different combination of known scalars:

$$C_i = Q_i + k_i \cdot G = (d_i + k_i) \cdot G, \quad (10)$$

where k_i is a known scalar chosen by the adversary. This shifts each unknown by a known amount.

Phase 2—Linear Combination into a Master Key. Define known coefficients $\{\alpha_i\}$ and compute:

$$C_{\text{master}} = \sum_i \alpha_i C_i = \left(\sum_i \alpha_i (d_i + k_i) \right) \cdot G. \quad (11)$$

Phase 3—Quantum Computation. Solve the ECDLP for C_{master} to obtain:

$$c_{\text{master}} = \sum_i \alpha_i d_i + \sum_i \alpha_i k_i \pmod{q}. \quad (12)$$

Since $\sum_i \alpha_i k_i$ is known, the adversary recovers $\sum_i \alpha_i d_i \pmod{q}$.

Phase 4—Repeated Aggregation with Independent Coefficients. Repeat Phases 2–3 with n linearly independent coefficient vectors $\{\alpha^{(1)}, \dots, \alpha^{(n)}\}$ to obtain n independent linear equations in the n unknowns $\{d_i\}$.

Remark 4.2. *This still requires n quantum ECDLP solutions—one per coefficient vector. The reduction to fewer quantum operations requires exploiting structural relationships among the private keys.*

4.4 Inversion and Key Recovery

The challenge of reducing n quantum operations to fewer is addressed through the following key insight.

Theorem 4.3 (MAKA Reduction). *If the adversary can identify or construct algebraic relationships among the target private keys (e.g. keys derived from a common seed, keys sharing structural properties, or keys within a hierarchical deterministic wallet), the effective number of independent unknowns is reduced.*

We analyse four cases in order of increasing power.

4.4.1 Case 1: Fully Independent Keys (Lower Bound)

When all n private keys are algebraically independent over $\mathbb{Z}_q$, the system $\mathbf{c} = \mathbf{A}\mathbf{d}$ requires exactly n independent values of $\mathbf{c}$ to solve. This means n ECDLP instances are necessary, and MAKA provides no asymptotic improvement in quantum operations. However, MAKA still enables:

- **Batched quantum processing** with structured input.
- **Classical parallelism** in pre-computation.
- **Error propagation analysis**—cracking any subset reveals partial information.

4.4.2 Case 2: Hierarchically Derived Keys (HD Wallets)

In Bitcoin’s BIP-32 Hierarchical Deterministic (HD) wallet scheme [4], child keys are derived as

$$d_i = d_{\text{parent}} + H(\text{chain_code}, i) \pmod{q}. \quad (13)$$

If the adversary knows (or can deduce) the derivation indices and chain code from public information, then all n keys reduce to **one unknown** (d_{parent}), and a single MAKA-aggregated quantum ECDLP cracks all keys simultaneously.

4.4.3 Case 3: Structured Key Generation (k unknowns, $k < n$)

More generally, if private keys are generated by a process with $k < n$ independent parameters, MAKA reduces the quantum workload to k ECDLP instances. The matrix construction is tailored to exploit the known structure:

$$\mathbf{d} = \mathbf{B}\mathbf{s} + \mathbf{t}, \quad (14)$$

where $\mathbf{s}$ is a k -dimensional vector of independent secrets, $\mathbf{B}$ is a known $n \times k$ matrix encoding the key-derivation structure, and $\mathbf{t}$ is a known offset vector. Substituting into the MAKA equation:

$$\mathbf{c} = \mathbf{A}(\mathbf{B}\mathbf{s} + \mathbf{t}) = (\mathbf{A}\mathbf{B})\mathbf{s} + \mathbf{A}\mathbf{t}. \quad (15)$$

The adversary computes $\mathbf{A}\mathbf{t}$ classically and obtains $(\mathbf{A}\mathbf{B})\mathbf{s}$. Since $\mathbf{A}\mathbf{B}$ is $n \times k$ and $\mathbf{A}$ is chosen freely, the adversary selects $\mathbf{A}$ to maximise the rank of $\mathbf{A}\mathbf{B}$, requiring only k quantum ECDLP solutions.

4.4.4 Case 4: The Injection Method (Novel Contribution)

The core novel idea is to construct the matrix system such that the unknown private keys interact multiplicatively (in the scalar field) with known points in a way that creates dependent equations.

Consider the construction where we form two matrices—one containing a mix of known and unknown scalars, the other containing only known scalars—and multiply them:

$$\mathbf{M} = \begin{pmatrix} k_1 & d_1 \\ k_2 & d_2 \end{pmatrix} \begin{pmatrix} k_3 & k_4 \\ k_4 & k_5 \end{pmatrix} = \begin{pmatrix} k_1k_3 + d_1k_4 & k_1k_4 + d_1k_5 \\ k_2k_3 + d_2k_4 & k_2k_4 + d_2k_5 \end{pmatrix}. \quad (16)$$

Each entry of the product matrix $\mathbf{M}$ is a known linear combination involving the unknowns. Because matrix multiplication creates **coupled** equations, the entries of $\mathbf{M}$ are not independent—they share the same unknowns combined differently with known values.

Mapping each entry back to a curve point:

$$M_{ij} \cdot G = (\text{linear combination of known and unknown scalars}) \cdot G.$$

The adversary computes each $M_{ij} \cdot G$ classically from the public keys and known points.

Specifically, for a 2×2 system with 2 unknowns (d_1, d_2) and 5 known values (k_1 through k_5): The product matrix yields four entries, each being a linear equation in d_1 and d_2 :

$$M_{11} = k_1k_3 + d_1k_4, \quad (17)$$

$$M_{12} = k_1k_4 + d_1k_5, \quad (18)$$

$$M_{21} = k_2k_3 + d_2k_4, \quad (19)$$

$$M_{22} = k_2k_4 + d_2k_5. \quad (20)$$

Cracking **any one** entry via quantum ECDLP gives a linear equation in one unknown:

- M_{11} or $M_{12} \Rightarrow$ yields d_1 directly.
- M_{21} or $M_{22} \Rightarrow$ yields d_2 directly.

But we can do better. **Sum all entries:**

$$\begin{aligned} S &= M_{11} + M_{12} + M_{21} + M_{22} \\ &= \underbrace{(k_1k_3 + k_1k_4 + k_2k_3 + k_2k_4)}_K + d_1(k_4 + k_5) + d_2(k_4 + k_5) \\ &= K + (k_4 + k_5)(d_1 + d_2), \end{aligned} \quad (21)$$

where K is fully known. A single ECDLP on $S \cdot G$ yields $(d_1 + d_2)$.

Similarly, a **weighted sum** with different coefficients yields a second independent equation. Two such sums $\Rightarrow$ two equations $\Rightarrow$ solve for both d_1 and d_2 , requiring only **2 ECDLP solves for 2 unknowns** but with the matrix structure providing a systematic framework for choosing optimal weightings.

The power of MAKa lies not in reducing below n quantum operations for fully independent keys (which information theory prohibits), but in:

1. Providing a **systematic algebraic framework** for constructing optimal aggregation strategies.
2. Exploiting **structural dependencies** in real-world key generation.
3. Enabling **partial recovery**—if any subset of composite keys is cracked, the matrix structure maximises the information extracted about the remaining unknowns.

5 Algebraic Analysis

5.1 Validity Over Elliptic Curve Groups

The MAKa construction requires that scalar-level linear algebra corresponds faithfully to curve-level point operations.

Proposition 5.1. *For points on an elliptic curve $E(\mathbb{F}_p)$ within the cyclic subgroup $\langle G \rangle$ of prime order q , and scalars $\alpha, \beta \in \mathbb{Z}_q$:*

$$\alpha(\beta G) = (\alpha\beta \bmod q) \cdot G, \quad (22)$$

$$\alpha G + \beta G = ((\alpha + \beta) \bmod q) \cdot G. \quad (23)$$

Proof. This follows directly from the definition of scalar multiplication as repeated group addition and the fact that $\langle G \rangle \cong \mathbb{Z}_q$ as groups. $\square$ $\square$

Corollary 5.2. *The MAKa construction—which involves only scalar multiplication of curve points and point addition—is algebraically valid. The composite points lie in $\langle G \rangle$ and correspond to well-defined scalar multiples of G .*

5.2 Invertibility Conditions

For the matrix equation $\mathbf{c} = \mathbf{A}\mathbf{d} \pmod{q}$ to be solvable for $\mathbf{d}$ given $\mathbf{c}$, the matrix $\mathbf{A}$ must be invertible over $\mathbb{Z}_q$.

Theorem 5.3 (Invertibility over $\mathbb{Z}_q$). *A matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ is invertible over $\mathbb{Z}_q$ if and only if $\gcd(\det(\mathbf{A}), q) = 1$. Since q is prime (as required for ECDSA-standard curves), this simplifies to $\det(\mathbf{A}) \not\equiv 0 \pmod{q}$.*

Proof. $\mathbb{Z}_q$ is a field when q is prime. Over a field, a matrix is invertible if and only if its determinant is non-zero. $\square$ $\square$

Since the adversary chooses $\mathbf{A}$ freely, ensuring invertibility is trivial—a random matrix over $\mathbb{Z}_q$ is invertible with probability $(1 - 1/q)^n \approx 1$ for large q .

Efficient Inversion. The inverse $\mathbf{A}^{-1}$ can be computed classically via Gaussian elimination in $\mathcal{O}(n^3)$ operations over $\mathbb{Z}_q$, or via Strassen’s algorithm in $\mathcal{O}(n^{2.807})$ operations. Once $\mathbf{c}$ is known (from quantum ECDLP), recovery is:

$$\mathbf{d} = \mathbf{A}^{-1}\mathbf{c} \pmod{q}. \quad (24)$$

5.3 Handling Degeneracies and the Point at Infinity

If any linear combination in the MAKa construction yields the *point at infinity* $\mathcal{O}$ (the identity element), the corresponding composite scalar is $0 \pmod{q}$, which is trivially known without quantum computation. This is a degenerate but harmless case—it simply means one equation is obtained “for free.”

The probability of accidentally hitting $\mathcal{O}$ is $1/q \approx 2^{-256}$ for standard curves—negligible.

6 Complexity Analysis

6.1 Baseline: Independent Quantum Attacks

Without MAKa, cracking n independent ECDSA keys requires:

Table 1: Baseline quantum cost for n independent keys.

Operation	Cost per key	Total (n keys)
Quantum ECDLP (Shor’s)	$\mathcal{O}(\log^2 q \cdot \log \log q)$	$\mathcal{O}(n \log^2 q \cdot \log \log q)$
Classical verification	$\mathcal{O}(\log q)$	$\mathcal{O}(n \log q)$
Total quantum cost	—	$\mathcal{O}(n \cdot \text{poly}(\log q))$

Table 2: MAKA-assisted cost breakdown.

Phase	Operation	Cost
Pre-processing	Matrix construction	$\mathcal{O}(n^2)$ scalar mults
Pre-processing	Composite points	$\mathcal{O}(n^2 \log q)$ EC ops
Quantum	ECDLP on composites	$\mathcal{O}(k \cdot \text{poly}(\log q))$
Post-processing	Matrix inversion	$\mathcal{O}(n^{2.807})$ field ops
Post-processing	Key recovery	$\mathcal{O}(n^2)$ field ops
Total		$\mathcal{O}(k \cdot \text{poly}(\log q) + n^2 \log q)$

6.2 MAKA-Assisted Complexity

With MAKA (assuming k effective unknowns after exploiting key structure):

6.3 Asymptotic Comparison

Table 3: Speedup under different key-structure scenarios.

Scenario	Baseline	MAKA	Speedup
Independent ($k=n$)	$\mathcal{O}(n \cdot \text{poly})$	$\mathcal{O}(n \cdot \text{poly})$	$1 \times$
HD wallet ($k=1$)	$\mathcal{O}(n \cdot \text{poly})$	$\mathcal{O}(\text{poly})$	$n \times$
Structured (k unkn.)	$\mathcal{O}(n \cdot \text{poly})$	$\mathcal{O}(k \cdot \text{poly})$	$n/k \times$
Shared params	$\mathcal{O}(n \cdot \text{poly})$	$\mathcal{O}(r \cdot \text{poly})$	$n/r \times$

The **maximum theoretical speedup** is $n \times$ (reducing n quantum operations to 1), achieved when all keys are derivable from a single secret. The classical overhead of MAKA is polynomial in n and dominated by the quantum phase for realistic parameters.

7 Worked Example

We illustrate MAKA with a simplified example using small numbers for clarity. In practice, all values would be 256-bit integers.

Setup. Curve: `secp256k1` (simplified for exposition); generator: G (standard base point); curve order: q (a large prime).

Target keys (unknown private keys).

$$d_1 = ? \rightarrow Q_1 = d_1 G, \quad d_2 = ? \rightarrow Q_2 = d_2 G, \quad d_3 = ? \rightarrow Q_3 = d_3 G.$$

Known scalars (chosen by adversary). $k_1 = 7, k_2 = 13, k_3 = 19$.

Step 1: Construct mixing matrix $\mathbf{A}$ (known, invertible over $\mathbb{Z}_q$).

$$\mathbf{A} = \begin{pmatrix} 3 & 5 & 2 \\ 7 & 1 & 4 \\ 9 & 8 & 6 \end{pmatrix}, \quad (25)$$

$$\det(\mathbf{A}) = 3(6 - 32) - 5(42 - 36) + 2(56 - 9) = -78 - 30 + 94 = -14 \not\equiv 0 \pmod{q}. \checkmark$$

Step 2: Compute composite points (classical, public keys only).

$$\begin{aligned} C_1 &= 3Q_1 + 5Q_2 + 2Q_3 = (3d_1 + 5d_2 + 2d_3) G, \\ C_2 &= 7Q_1 + 1Q_2 + 4Q_3 = (7d_1 + d_2 + 4d_3) G, \\ C_3 &= 9Q_1 + 8Q_2 + 6Q_3 = (9d_1 + 8d_2 + 6d_3) G. \end{aligned}$$

Step 3: Inject known points (shift by known scalars).

$$\begin{aligned} C'_1 &= C_1 + k_1 G = (3d_1 + 5d_2 + 2d_3 + 7) G, \\ C'_2 &= C_2 + k_2 G = (7d_1 + d_2 + 4d_3 + 13) G, \\ C'_3 &= C_3 + k_3 G = (9d_1 + 8d_2 + 6d_3 + 19) G. \end{aligned}$$

Step 4: Quantum ECDLP on each C'_i yields:

$$\begin{aligned} c'_1 &= 3d_1 + 5d_2 + 2d_3 + 7, \\ c'_2 &= 7d_1 + d_2 + 4d_3 + 13, \\ c'_3 &= 9d_1 + 8d_2 + 6d_3 + 19. \end{aligned}$$

Step 5: Subtract known offsets.

$$c_1 = c'_1 - 7, \quad c_2 = c'_2 - 13, \quad c_3 = c'_3 - 19.$$

Step 6: Solve $\mathbf{A}\mathbf{d} = \mathbf{c}$ via matrix inversion.

$$\mathbf{d} = \mathbf{A}^{-1}\mathbf{c} \pmod{q}.$$

This recovers all three private keys d_1, d_2, d_3 .

Remark 7.1 (Quantum operations). *This example still requires 3 quantum ECDLP solutions (one per composite key). The reduction to fewer quantum operations requires exploiting structural relationships among d_1, d_2, d_3 as described in Section 4.4.*

HD Wallet Variant. If the keys were derived as $d_i = d_{\text{master}} + \text{hash}(i)$, then:

$$c_1 = 3(d_m + h_1) + 5(d_m + h_2) + 2(d_m + h_3) = 10d_m + (3h_1 + 5h_2 + 2h_3).$$

The known hash offsets are subtracted, leaving a single unknown d_{master} . **One quantum ECDLP suffices** to recover all three private keys.

8 Limitations and Open Problems

8.1 Information-Theoretic Bound

Fundamental Limitation. For n truly independent private keys, each uniformly random in $\mathbb{Z}_q$, the joint entropy is $n \cdot \log_2 q$ bits. A single ECDLP solution provides at most $\log_2 q$ bits of information. Therefore, **at least n quantum ECDLP solutions are information-theoretically necessary** for fully independent keys.

MAKA cannot circumvent this bound. Its value lies in:

- Exploiting **non-independence** in real-world key generation.
- Providing a **systematic framework** for optimal aggregation.
- Enabling **partial recovery** with fewer than n quantum operations.

8.2 Practical Considerations

1. **Curve Point Computation Cost.** Computing the composite points requires $\mathcal{O}(n^2)$ elliptic curve scalar multiplications, each costing $\mathcal{O}(\log q)$ group operations. For $n = 10^6$ keys on a 256-bit curve, this is $\sim 10^{12} \cdot 256 \approx 2.56 \times 10^{14}$ group operations—substantial but feasible with classical parallelism.
2. **Matrix Size.** For very large n , the mixing matrix $\mathbf{A}$ has n^2 entries in $\mathbb{Z}_q$. Storage is $n^2 \cdot 32$ bytes. For $n = 10^6$, this is ~ 32 TB—large but within the capacity of modern storage systems.
3. **Numerical Stability.** All operations are exact (modular arithmetic over $\mathbb{Z}_q$), so there are no floating-point stability concerns. However, the matrix inversion must be performed exactly over $\mathbb{Z}_q$.

8.3 Open Problems

1. **Optimal Matrix Design.** What structure of $\mathbf{A}$ maximises information extraction per quantum ECDLP when only a subset of composite keys can be cracked? This connects to coding theory and compressed sensing.
2. **Non-linear Aggregation.** Can non-linear combinations of public keys (e.g. using pairings on elliptic curves) provide advantages over linear MAKAs?
3. **Quantum–Classical Hybrid Optimisation.** Can the classical matrix pre-processing be interleaved with quantum operations to adaptively reduce the number of required ECDLP solutions?
4. **Extension to Other Cryptosystems.** Does the MAKAs framework apply to other discrete-log-based systems (e.g. EdDSA, Schnorr signatures, Diffie–Hellman key exchange)?
5. **Multi-Curve Aggregation.** Can keys from different elliptic curves be combined via MAKAs using curve isogenies or other inter-curve mappings?

9 Defensive Implications

The MAKAs framework, even in its theoretical form, has significant implications for cryptographic defence.

9.1 Key Generation Hygiene

- **Avoid deterministic key derivation** where the derivation structure is publicly inferrable. HD wallets should use hardened derivation paths that include private-key material in the derivation function.
- **Inject additional entropy** at each key-generation step to ensure true independence.
- **Rotate keys frequently** to limit the number of keys an adversary can aggregate.

9.2 Post-Quantum Migration

- **Urgency of PQC adoption:** MAKAs amplifies the HNKL threat by showing that structured key collections are even more vulnerable than individual keys. This increases the urgency of migrating to post-quantum cryptographic algorithms (e.g. CRYSTALS-Dilithium, FALCON) standardised by NIST [5].
- **Hybrid schemes:** During the transition period, deploy hybrid signature schemes combining ECDSA with a post-quantum algorithm to ensure security against both classical and quantum adversaries.

9.3 Monitoring and Detection

- Organisations should inventory all ECDSA keys in use and assess whether their generation processes create exploitable algebraic relationships.
- Network monitoring should flag anomalous patterns of public-key collection that might indicate HNKL activity.

10 Related Work

Shor’s Algorithm [1]. The foundational quantum algorithm for discrete logarithms. MAKAs builds upon Shor’s as the underlying quantum primitive.

Proos and Zalka [3]. Provided detailed quantum resource estimates for ECDLP, establishing the concrete cost of breaking individual ECDSA keys.

Roetteler et al. [6]. Analysed quantum resource requirements for attacking elliptic curve cryptography, providing optimised quantum circuit designs.

Boneh and Shoup [7]. Comprehensive treatment of applied cryptography including batch attacks on discrete-log systems. The MAKa framework extends their batch DLP analysis to the matrix setting.

BIP-32 [4]. The Bitcoin Improvement Proposal for hierarchical deterministic wallets, whose key-derivation structure is a prime target for MAKa-style aggregation.

Bernstein and Lange [8]. Analysed post-quantum cryptographic alternatives and the timeline for quantum threats, providing context for the HNDL model.

Multi-Scalar Multiplication (MSM) [9]. Efficient algorithms for computing linear combinations of elliptic curve points, directly applicable to MAKa’s classical pre-processing phase. Techniques like Pippenger’s algorithm reduce the cost of MSM from $\mathcal{O}(n \log q)$ to $\mathcal{O}(n \log q / \log n)$.

11 Conclusion

This paper introduced **Matrix-Accelerated Key Aggregation (MAKA)**, a theoretical framework that leverages matrix multiplication over elliptic curve scalar fields to restructure the batch ECDLP problem faced by quantum adversaries in Harvest Now, Decrypt Later scenarios.

Our key contributions are:

1. **Formalisation** of the matrix aggregation construction for ECDSA public keys, proving its algebraic validity over elliptic curve groups.
2. **Complexity analysis** showing that MAKa reduces quantum cryptanalytic cost from $\mathcal{O}(n \cdot \text{poly}(\log q))$ to $\mathcal{O}(k \cdot \text{poly}(\log q))$ where $k \leq n$ is the effective number of independent unknowns after exploiting key-generation structure.
3. **Identification of vulnerable key-generation patterns**, particularly hierarchical deterministic wallets, where MAKa enables recovery of all derived keys from a single quantum ECDLP solution.
4. **The Injection Method**, a novel technique for embedding known curve points into a matrix multiplication framework to create algebraically coupled systems of equations.
5. **Defensive recommendations** for key-generation practices and post-quantum migration strategies informed by the MAKa threat model.

While MAKa cannot break the information-theoretic lower bound of n quantum operations for n fully independent keys, real-world key generation rarely achieves true independence. The structured nature of practical key management—from HD wallets to key-derivation functions—creates algebraic relationships that MAKa systematically exploits.

The framework opens several avenues for future research, including optimal matrix design, non-linear aggregation via pairings, and extension to other discrete-log-based cryptosystems. Most importantly, MAKa underscores the urgency of post-quantum cryptographic migration: the threat to ECDSA is not merely per-key but potentially systemic.

Disclaimer

This paper presents a theoretical cryptanalytic framework for academic research purposes. The authors do not endorse or encourage the use of these techniques for unauthorised access to cryptographic systems. The primary intent is to inform defensive cryptographic practices and motivate the adoption of post-quantum cryptography.

References

- [1] P.W. Shor, “Algorithms for quantum computation: Discrete logarithms and factoring,” in *Proc. 35th Annual Symp. Foundations of Computer Science (FOCS)*, pp. 124–134, 1994.

- [2] M. Mosca, “Cybersecurity in an era with quantum computers: Will we be ready?,” *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018.
- [3] J. Proos and C. Zalka, “Shor’s discrete logarithm quantum algorithm for elliptic curves,” *Quantum Information & Computation*, vol. 3, no. 4, pp. 317–344, 2003.
- [4] P. Wuille, “BIP-32: Hierarchical Deterministic Wallets,” *Bitcoin Improvement Proposals*, 2012. <https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki>
- [5] National Institute of Standards and Technology, “Post-Quantum Cryptography Standardization,” NIST, 2024. <https://csrc.nist.gov/projects/post-quantum-cryptography>
- [6] M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, “Quantum resource estimates for computing elliptic curve discrete logarithms,” in *Advances in Cryptology—ASIACRYPT 2017* (LNCS, vol. 10625), pp. 241–270, 2017.
- [7] D. Boneh and V. Shoup, *A Graduate Course in Applied Cryptography*, Version 0.6, 2023. <https://toc.cryptobook.us/>
- [8] D. J. Bernstein and T. Lange, “Post-quantum cryptography,” *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.
- [9] D. J. Bernstein, J. Doumen, T. Lange, and J.-J. Osterwijk, “Faster batch forgery identification,” in *Progress in Cryptology—INDOCRYPT 2012* (LNCS, vol. 7668), pp. 454–473, 2012.